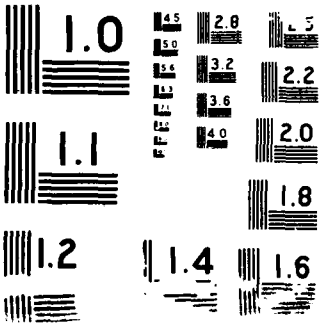
AD-A194 370 HIERARCHICAL ROUTE PLANNER(U) ARMY ENGINEER TOPOGRAPHIC 1/1
LABS FORT BELVOIR VA J R BENTON 28 JAN 88 ETL-R-139

UNCLASSIFIED F/G 17/7.1 NL


1.0
1.1
1.2
1.4
1.6
2.8
3.2
3.6
4.0
2.2
2.0
1.8

AD-A194 370 DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

UNCLASSIFIED

1b RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

3 DISTRIBUTION / AVAILABILITY OF REPORT

Approved for public release; distribution is unlimited.

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

4. PERFORMING ORGANIZATION REPORT NUMBER(S)

R-139

5. MONITORING ORGANIZATION REPORT NUMBER(S)

Accesion For
NTIS CRA&I
DTIC TAB
Unannounced
Justification
By
Distribution /
Availability Codes
Dist | Avail and / or Special

6a. NAME OF PERFORMING ORGANIZATION

USAETL

6b. OFFICE SYMBOL (If applicable)

CEETL-LO

7a. NAME OF MONITORING ORGANIZATION

6c. ADDRESS (City, State, and ZIP Code)

Fort Belvoir, VA 22060-5546

7b. ADDRESS (City, State, and ZIP Code)

8a. NAME OF FUNDING / SPONSORING ORGANIZATION

8b. OFFICE SYMBOL (If applicable)

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS (City, State, and ZIP Code)

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
		A-1	

11. TITLE (Include Security Classification)

HIERARCHICAL ROUTE FINDER

12. PERSONAL AUTHOR(S)

JOHN R. BENTON

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
RESEARCH	FROM _____ TO _____	88 JANUARY 28	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	mobility, automated terrain reasoning, Lisp, route planning, autonomous vehicles

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

A two-level hierarchical route planner has been develped. The data input to the system is a cross-country mobility map. For a given vehicle type, this map specifies regions which are "GO" or "NO-GO." A line-thinning algorithm is used to generate a skeleton of the "GO" areas. This skeleton is then converted into a graph-theoretic structure. A first-level route planner using elevation-grid data is used to compute the traversal time of each arc of the graph. These traversal times become the weights used by the second level route planner. This route planner is an A* algorithm that is used to search for a specified number of non-competing routes, i.e., routes that have no arc-segments in common. Thus, the first level route planner does detailed planning over a small area but is subject to combinatorial explosion when a search over a wider area is required. The second level graph-search algorithm provides the capability to efficiently plan a route over a larger area but without detail about the precise path followed. This system was implemented in Common Lisp on a Lisp machine. The software has also been integrated into a workstation that was developed to provide

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT

☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION

UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
F. DARLENE SEYLER	(202) 355-2647	CEETL-LO

DD Form 1473, JUN 86 Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Block #19

support to Army robotic vehicle research. The workstation provides support for comparing the capabilities of alternative route finding algorithms.

HIERARCHICAL ROUTE PLANNER

John R. Benton

U.S. Army Engineer Topographic Laboratories
Fort Belvoir, Virginia 22060-5546

ABSTRACT

A two-level hierarchical route planner has been developed. The data input to the system is a cross-country mobility map. For a given vehicle type, this map specifies regions which are "GO" or "NO-GO." A line-thinning algorithm is used to generate a skeleton of the "GO" areas. This skeleton is then converted into a graph-theoretic structure. A first-level route planner using elevation-grid data is used to compute the traversal time of each arc of the graph. These traversal times become the weights used by the second level route planner. This route planner is an A* algorithm that is used to search for a specified number of non-competing routes, i.e., routes that have no arc-segments in common. Thus, the first level route planner does detailed planning over a small area but is subject to combinatorial explosion when a search over a wider area is required. The second level graph-search algorithm provides the capability to efficiently plan a route over a larger area but without detail about the precise path followed. This system was implemented in Common Lisp on a Lisp machine. The software has also been integrated into a workstation that was developed to provide support to Army robotic vehicle research. The workstation provides support for comparing the capabilities of alternative route finding algorithms.

1. INTRODUCTION

Anyone who has struggled to find the most direct route between two locations on a city map where there is no single road or even just two or three roads directly connecting the two locations is familiar with the general problem of planning an optimum path through a complicated tangle of roads. In planning a route through a city one must consider not only distance but also the number of traffic lights and quantity of traffic at a given time of day. A battlefield commander may not have to worry about traffic lights when planning a route, but he must consider positions of enemy and friendly forces, lines of sight from enemy observation points, availability of places of concealment from aerial observation, choke points where his forces could be ambushed, and weather conditions that could affect the mobility of his vehicles.

Currently, there is no capability of automated route finding available for operational use in the Army. Such a capability would be particularly useful in situations that require a rapid reaction to counter an enemy threat or changing conditions. One example of an Army application where this capability would be valuable is the requirement that a friendly force's commander be able to predict the enemy's objective and the routes the enemy will use to advance toward their objective. Military doctrine often calls for offensive forces to advance in three separate columns. The defending forces must use their knowledge of the doctrine and tactics of the opposing force as well as knowledge of the terrain to anticipate and possibly to counteract the movement of their enemy. Terrain analysis performed in support of battlefield planning is tedious and time consuming. A detailed terrain analysis of a strategically important area may take as long as a month to perform. It is also well known that accurate knowledge of the battlefield is critical to successful operations. Since it is not feasible to analyze in advance all potentially important areas, automated terrain reasoning provides a possible way out of this bind. One of the tasks that could be performed by the computer system would be to predict enemy movements. As mentioned above, military plans will frequently involve three separate columns. The computer should therefore compute three optimum noncompeting paths as well as suboptimum alternatives. The three routes must be noncompetitive (i.e. have no road segments in common) since traffic jams would be engendered if two separate columns had to share the same road.

To address these problems, a project was initiated to develop a system capable of automatically generating optimized routes for multiple tank columns. Although this project is aimed primarily at developing a potential to provide battlefield support,

this work is also directly applicable to route planning for autonomous vehicles.

2. ANALYSIS

All of the information required to determine the paths to an objective may be present in a cross-country-mobility (CCM) map. However, this does not mean that it will be easy for even a trained analyst to quickly determine multiple optimum routes. Since the allowable velocity along a selected path can vary according to the type of terrain encountered, the only way that the traversal time can be computed is to integrate the velocity along the path.

In developing a computer model for route planning over a large area, a prime concern must be to avoid what is known as "combinatorial explosion." For example, a four-levels-deep full binary tree has 15 branch points or nodes while a 20-levels-deep full binary tree has more than a million nodes. If the map data consist of a grid at 10 meter spacings, an unrestricted search algorithm would generate a search tree with literally millions of nodes* in order to plan a route for a distance of less than 200 meters. Obviously, no one would actually use such a simple-minded technique, but when route planning over a distance of several kilometers is required, the almost inevitable result of planning a route at the pixel level is combinatorial explosion. An alternative approach is to reduce the size of the search space before the planning algorithm is invoked.

There are several methods that can be used to reduce the search space. However, they can be divided into two classes: (1) preprocessing methods and (2) methods for searching more intelligently. The preprocessing methods are concerned with the data representation of a mobility map. The maps are normally stored in one of two forms in a computer, raster and vector. The raster is simply the ordered rows of pixels mentioned above. The vector representation makes use of the fact that typically a large area of the map has identical mobility factors. This area can be at least approximately bounded by a polygon of contiguous vectors. The area, or "region" as it is generally called, can then be represented in the computer by a list of the vectors of which the polygon is composed. In contrast to the region representation, each pixel is surrounded by either four or eight neighboring pixels, depending on the definition of neighbor that is used. Thus from a given node, the number of neighbors and the distance to each neighbor are always constant. In contrast, with a polygon representation, the number of neighbors will vary, and the distance to each neighbor is no longer well defined since the shape of the regions may be irregular. A method intermediate in storage efficiency uses a quadtree representation[1] in which the map of a square area is divided into four quadrants. Each quadrant is subdivided if the quadrant is not completely contained within a uniform region. The process terminates when no regions remain to be subdivided. The distance from the center of any quadrant to the center of an adjacent quadrant (which may have been subdivided a different number of times and therefore is a different size) is easily computed.

There is one additional representation method which can result in a large reduction in storage requirements. Any one of a number of line-thinning algorithms can be applied to the mobility map in order to generate a skeletal structure that will correspond to a line drawn along the center of mobility corridors or avenues of approach. Branches in the skeletal structure are caused by obstacles that force a traveler to fork either to the left or to the right in order to go around the obstacle. Since the line-thinning algorithms require a binary image, the first step is to generate a mobility map containing only two levels of mobility: go or no-go. An illustration of this line-thinning technique is shown in Figure 1. The shaded figure represents a region, with the skeleton indicated by the superimposed asterisks. This skeleton can also be considered to be a graph-theoretic structure to which graph theory can be applied. A tree-searching algorithm can be used to find the optimum paths through the graph. However, the lines of the graph generated by the line-thinner may not be the optimum path between those two nodes. The solution is to use a pixel-level route planner to generate the precise route between nodes and a graph level route planner to efficiently plan a route over a larger area but without detail about the precise path that was followed. The graph-level planner uses the

Figure 1. Example of line thinning algorithm used to generate stick figure.

* A node indicates a point along a path where there is a choice in which direction to proceed.

traversal times generated by the pixel-level planner. The resulting system is a two-level hierarchical route planner. This is the approach that was selected for use on this project. Intelligent search methods can be applied at both route planner levels.

3. IMPLEMENTATION

The overall system is shown in Figure 2 and is being implemented in Common Lisp on a Symbolics Lisp Machine. The Geographic Information System (GIS) will contain road data in vector format and will be able to generate CCM maps in raster format. Figure 3 is a simplified map with "GO" areas shown in white and "NO-GO" areas shown in black. Superimposed on this CCM map is the vectorized skeleton obtained after line-thinning the "GO" area and performing a raster to graph conversion. A comparison of the graph to the CCM map clearly indicates that in several places, two closely spaced nodes should be merged into one since the mobility map shows that four paths come together into a common junction while the corresponding place on the graph contains two closely spaced nodes. The node-merger module merges closely spaced nodes and generates the graph shown in Figure 4. Finally, Figure 5 shows the results of asking the graph-level route planner to determine three paths from node 1 to node 35. The solid dark lines indicate the shortest route, the dotted lines show the next shortest path and the dashed lines represent the longest of the three paths. The nodes that are not labeled in the figure were explored during the search

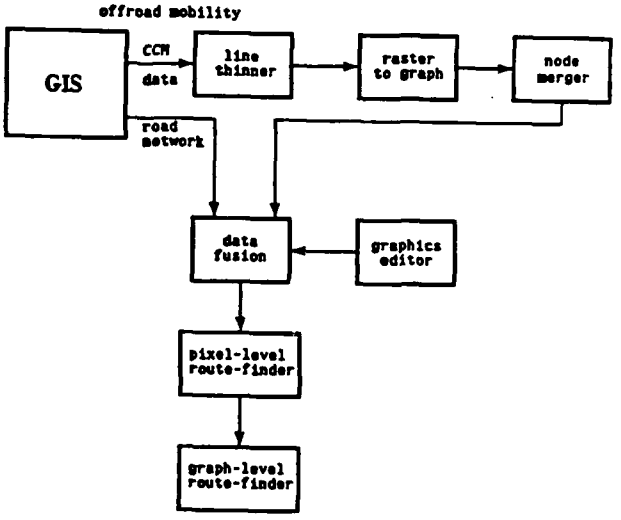


Figure 2. Hierarchical Route Planner data flow diagram.

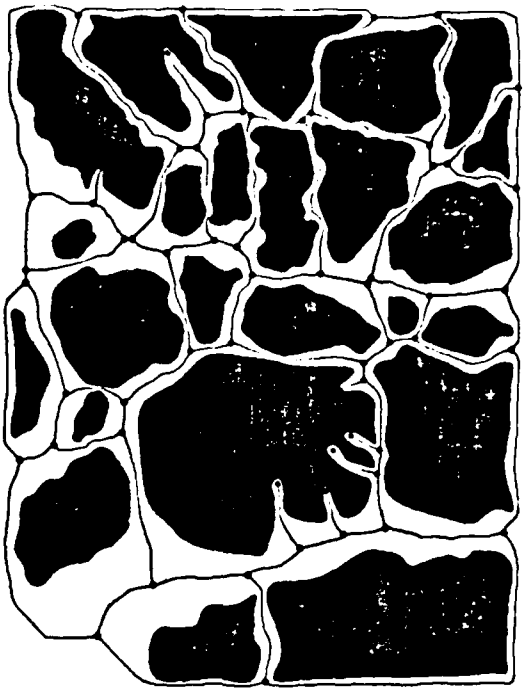


Figure 3. Cross-Country-Mobility Map with skeleton of "GO" areas superimposed. Black represents obstacles.

for the three optimum routes. The direct route shown by the path 1-3-5-35 could actually include a mountain pass and require a longer time to traverse than the path 1-10-8-7-6-55-54-36-35 which is much longer in distance. The weight attached to each arc of the graph determines which case is actually optimum in terms of traversal time.

3.1. Geographic Information System

The QUILT Geographic Information System[2] has been selected for use on this project. It is quad-tree based, but with b-trees used as the underlying storage mechanism in order to increase the efficiency in accessing very large sets of data. QUILT also contains a relational query system. QUILT contains very efficient mechanisms for the rapid computation of intersection and union operations of two map overlays. QUILT is also being used on the Expert System for Minefield Site Prediction which is being developed under contract to USAETL. QUILT will provide the capability to convert CCM maps into the raster format required by the line thinner module. Road network information will also be extracted from QUILT. QUILT was recently installed on a VAX 780 computer with Grinnell display and on a Sun Workstation using

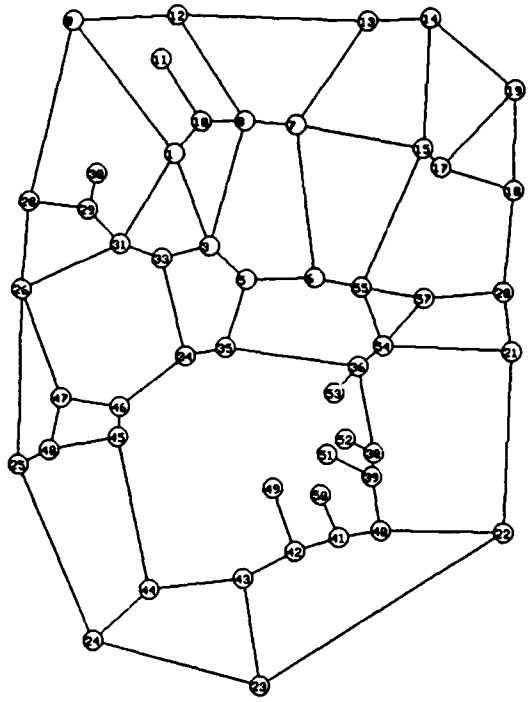


Figure 4. Graph representation of CCM map after merger of closely adjacent nodes.

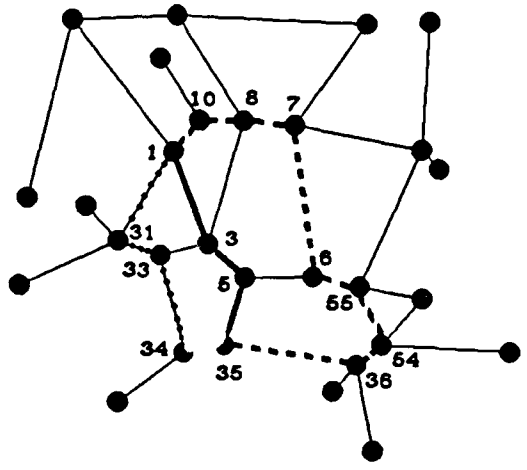


Figure 5. Three optimum routes of path from node 1 to node 35.

the newly released Version 11 X Windows[3] package. When an X Windows package becomes available on the Symbolics Lisp Machine, it will be possible to run QUILT on the VAX or Sun and display the map information on the Lisp machine since X Windows provides this networking capability.

3.2. Line Thinning

A line thinning algorithn developed by Chang and Suen[4] was implemented in Common Lisp on a Lisp machine. This algorithm was designed so that all scan lines can be thinned simultaneously on a parallel machine. A 500 by 700 pixel raster currently requires 20 minutes to run on a Symbolics 3600 computer. It should be possible to reduce the time to about one minute by adding a list of zero-to-one transitions for each raster line and also a flag for each scan line indicating whether any points were eliminated on that line on the previous iteration.

3.3. Raster To Graph

This module accesses the screen array and scans the array until it finds the first non-zero pixel. It then looks at the eight neighbors of the pixel and traces along the line until it finds a pixel that has two neighbors which are not adjacent. This pixel becomes node 0, and its value is changed to some other non-zero value so that it will have a different color on the display and more importantly be invisible to the line following algorithm. Chain coding is used to store the paths, with values from zero to seven used to represent the eight possible directions. As the line follower advances along the line, each pixel has its value changed so that the line follower will not circle back on itself. Tests are also performed to detect cases where the current pixel encounters three adjacent pixels arranged along a straight line or the case where the current pixel and the three adjacent neighbors form a square. Whenever the path follower is confronted with choosing between pixels adjacent to each other, the principal axis direction is chosen in preference to a diagonal direction. This path follower correctly located the 60 nodes of the graph in Figure 3. All pixels of the graph were traversed. The outputs of this program are (1) the chain-codes of the arcs, (2) a sparse array containing the lengths of the chain-codes and the number designation of each connecting node, and (3) a one-dimensional array contains the (x,y) coordinates of each node.

3.4. Node Merger

This module scans all detected arcs and eliminates closely-spaced nodes. When four or more paths of the CCM map converge to a single junction, the area of the intersection is frequently quite large compared to the width of the paths. The line thinner often generates adjacent nodes which need to be merged into a single node. The node-merger module merges nodes if the separation of the nodes is less than

a threshold value. This value is currently a constant which is transferred to the node-merger function on the argument list. The threshold value used in generating Figure 4 was 25 pixels in vertical and horizontal directions. A total of 10 nodes were merged with adjacent nodes. A future enhancement will be to have the module check to see if both nodes actually are located at the same junction of the cross-country-mobility map.

3.5. Graphics Editor

The route planner program must provide a capability for allowing the user to modify the generated graph. Changes may be necessary due to new information obtained from aerial photographs, reconnaissance, or other data available to the user. The user interface of this module has not been completed but most of the functional capabilities required by this module have been coded. One sub-module allows the user to add nodes and arcs to the graph by tracing the arc-paths with a mouse pointer. Spline curves are fitted to the traced lines and the data structures required by the graph-level route planner are generated. When a path is being traced, if there is a sharp turn in the traced path, then a separate spline curve will be used on each side of the turn. A discontinuity in slope can thereby be accommodated. The route planner can be run using data generated solely by the mouse-tracker. Functions contained in the node-merger module will also be used in the graphics editor.

3.6. Data Fusion

When there is both road data in vector format from the GIS and graph derived from the CCM, a capability to merge road data will be required. Development of this module will not begin until the GIS has been integrated into the route planner. Functionally, this module will be similar to the raster-to-graph module.

3.7. Pixel Level Route Planner

A number of pixel-level route planners have been developed for use in the Route Planner Development Workstation[5] (RPDW). The RPDW software package was developed for USAETL through an agreement with NASA. The developers of this software system provided "hooks" to ease the porting of new route planners into the RPDW. One of the pixel-level route planners on this system will be used to determine the weights used in the graph-level route planner.

3.8. Graph-Level Route Planner

An algorithm essentially equivalent to the A* algorithm developed by Hart, Nilsson, and Raphael[6] was used to search the graph structure for an ordered listing of noncompetitive optimum routes between two given nodes. Each route is represented by a linked list of nodes, with the first node being the start and the final node being the destination node. The route with the lowest cost is listed first. Since the routes are noncompetitive, no two routes can share a path and thus they cannot share two successive nodes in their respective paths. The program is currently more restrictive in that no two paths can share a road intersection. Depending on the road widths at intersections, this may or may not be a necessary restriction.

The major difference of the Multiple Route Planner[7] (MRF) algorithm used here from A* is that A* halts as soon as the first optimum route is discovered, while MRF keeps searching until either there are no more unexplored routes or the required number of routes has been found. A* keeps two separate lists of nodes called OPEN and CLOSED. OPEN is the list of nodes that have not yet been explored. Initially, it contains only the start node. When a node is explored, it is moved from OPEN to CLOSED, and its descendants are put on the OPEN list. However, if one of these nodes has previously been reached by another route, then the more costly of the two routes must be pruned, and only the cheaper of the two routes will be retained. The MRF algorithm differs from A* in that the OPEN and CLOSED list are kept as a single linked list, with a slot at the node address to specify whether the node type is TERMINAL (CLOSED), NONTERMINAL (OPEN), PRUNED, or DESTINATION. A* terminates when the destination node is reached, in contrast to MRF, which continues to search for additional routes; therefore an explicit DESTINATION type is needed.

The A* algorithm uses a heuristic function to estimate the cost to go from the current node to the destination node. This function is usually called h(n), where n is the current node. The total estimated cost associated with the node n is $f(n) = g(n) + h(n,)$ where g(n) is the actual cost in traveling from the start node to the node n. Thus, f(n) is the estimated total cost for traversing a path from the start node to the point n and then on to the destination node. A heuristic function can be shown to be "admissible" if it is monotonically nondecreasing and nonnegative. If the heuristic function is admissible, then the algorithm is guaranteed to terminate with the optimum solution. Conceptually, the simplest heuristic that meets these requirement is the euclidean distance from the node n to the destination node. The actual cost can equal

but never be less than this straight line distance from n to the destination.

4. AUTOMATED TERRAIN REASONING

The Hierarchical Route Planner shown in Figure 2, combined with one or more expert system shells will provide the nucleus of an Automated Terrain Reasoning (ATR) Testbed. Additional components that will be added to the system in the near future include an environmental effects data base, and Automated Drainage Network Delineation[8] (ADND) software. ADND was done as part of a cooperative project with several USAETL laboratories providing personnel for the overall project. The input to ADND is a dense grid of elevation data. The program first extracts all saddle points which represent minima through which water could flow. The crucial elements in the program are the processes of converting this raw minima data in raster format into vector form, linking discontinuous data and structuring the vector data to reflect the ordering of the drainage branches. Future research will be in using the vectorized drainage data to automate the identification of the drainage pattern types. Using this derived information, it is then possible to make inferences about the types of soil and rock that underlie the region. This information is of critical importance in determining mobility and is often not easily available.

The ATR Testbed will provide support for future research on the interactions of weather, terrain analysis and military doctrine. Close liaison will be maintained with other Army laboratories and schools to identify areas of collaboration in which automating elements of terrain analysis will lead to increased effectiveness of the Field Army.

5. REFERENCES

1. H. Samet, "The Quadtree and Related Hierarchical Data Structures,"*ACM Computing Surveys*, **16** *2*, 187-260 (1984).

2. C. A. Shaffer, H. Samet, and R. Nelson, *QUILT, A Geographic Information System Based on Quadtrees*, Center for Automation Research, University of Maryland, College Park Maryland, 20742, Report No. CAR-TR-307 CS-TR-1885, (1987).

3. R. W. Schiefler and J. Gettys, "The X-Window System", *ACM Transactions on Graphics*, **5** *2*, 79-109 (1986).

4. T. Y. Zhang, and C. Y. Suen, "A Fast Parallel Algorithm for Thinning Digital Patterns," *Comm. of the ACM*, **27** *3*, 236-239 (1984).

5. J. Cameron, B. Cooper, A. Mishkin, K. Holmes, *Route Planner Development Workstation - Functional Description*, Jet Propulsion Laboratory, Pasadena, California, Report JPL D-2733 Vol. 1, (1985).

6. P.E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", *IEEE Trans. System Science and Cybernetics*, **SSC-4** *2*, 100-107 (1968).

7. J. R. Benton, *Automated Route Finder for Multiple Tank Columns*, U.S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia (1987).

8. W. W. Seemuller, "The Extraction Of Ordered Vector Drainage Networks From Elevation Data", (Submitted for Publication in Technical Journal), (1988).

END

DATE

FILMED

8-88

DTIC